

专业课程设计I报告

（ 2019 / 2020学年 第 二 学期）

题 目： 文件完整性检测系统

|  |  |
| --- | --- |
| **专 业** | **信息安全** |
| **学 生 姓 名** | **徐雪冉** |
| **班 级 学 号** | **B17040907** |
| **指 导 教 师** | **曹晓梅** |
| **指 导 单 位** | **信息安全系** |
| **日 期** | **2020.06.08-2020.06.18** |

|  |  |  |
| --- | --- | --- |
| **支撑指标点** | **评价准则** | **计分（每项10分）** |
| 课程目标1：通过课程设计，培养学生综合应用密码学、计算机技术等领域专业知识的技能，养成良好的职业道德。（20分） | 1、能够掌握密码学的相关基础知识，并能够针对求解的工程问题，收集资料进行合理的分析与设计。。 |  |
| 2、具备一定自学能力与探索创新意识，能够充分利用教科书及其资源（如网络等）自学新知识与新技能。 |  |
| 课程目标2：培养学生解决信息安全领域复杂工程问题的实践创新能力。（20分） | 3、通过调研，能够选择合适的程序设计语言与编程开发平台，对求解的工程问题进行编程实现。 |  |
| 4、具备一定的人机交互设计意识，人机交互设计合理、友好，操作简便。 |  |
| 课程目标3：提高学生文献查阅与资料收集能力，发现问题、研究问题、分析问题与解决问题的能力。（10分） | 5、能够对实际工程问题进行形式化描述，给出数据结构和算法的设计描述，给出关键算法的流程图或伪代码，并给出各算法之间的结构关系描述。 |  |
| 课程目标4：密码学相关的工程基础实验验证与实现能力，能够根据要求设计实验方案并开展实验，能够对实验数据进行解释与对比分析，给出实验结论。（20分） | 6、能够结合计算机软硬件资源，合理选用算法、数据结构、数据存储方式等技术手段，理解相关算法，能够完成课题要求的各项任务和指标。 |  |
| 7、掌握调试方法与工具，对程序开发过程中出现的问题进行分析、跟踪与调试，并能够进行充分测试。 |  |
| 课程目标5：分组完成一次课程设计与开发的全过程，组内成员通过讨论和交流解决实验中的难题，能在实验报告中准确阐述实验内容，能够在答辩时清晰陈述观点和回答问题。（30分） | 8、组内成员之间有一定的团队合作，互通有无。 |  |
| 9、具备一定的语言表达能力与文字处理能力，能够结合复杂工程问题撰写报告，报告内容和实验数据详实，格式规范。 |
| 10、能够正确、完整地回答指导教师关于课题的问询，反映其对课题内容，以及相关的工程基础知识具有较好的理解和掌握。 |
| 专业课程设计I能力测评总分 | |  |
| **指导教师： 年 月 日** | | |
| **备注：** | | |

**文件完整性检测系统**

**一、课题内容和要求**

在网络空间，文件可能因为感染病毒、植入木马/后门、人为篡改或者传输故障等原因导致被篡改。本题目致力于设计和实现一款文件完整性检测系统，通过CRC32、MD5或SHA-1等算法来比对确定文件的状态，从而可以及时检测出文件的异常改动，使学生深入了解完整性校验算法的实现原理，提升编程能力。

课题要求：

1. 查阅相关资料，掌握CRC32、MD5或SHA-1等完整性校验算法的原理和实现细节；

(2) 掌握Windows系统不少于两种类型文件（如DLL、DRV、SYS、OCX、EXE、COM、

PIF、SCR等）的获取、结构特点及散列值的计算和比较；

1. 选择一种编程语言和开发工具，设计和实现系统文件完整性检测系统，对不同类型、不同大小的文件进行完整性验证，分析比较检测的准确率、执行效率等指标；
2. 程序具有图形化用户界面，输出美观；
3. 可根据自己能力，在完成以上基本要求后，对程序功能进行适当扩充；
4. 撰写报告，对所采用的算法、程序结构和主要函数过程以及关键变量进行详细的说明；对程序的调试过程所遇到的问题进行回顾和分析，对测试和运行结果进行分析；总结软件设计和实习的经验和体会，进一步改进的设想；
5. 提供关键程序的清单、源程序及可执行文件和相关的软件说明。

**二、课题需求分析**

本课题目标系统“文件完整性检测系统”的功能框架图如图1所示。

开始

获取文件路径

以字节流的方式读入文件

计算文件的散列值

将两个文件的散列值进行对比

散列值是否相等

否

文件被修改

文件未被修改

退出程序

结束

图1 功能框架图

1. 支持系统根据文件路径导入两个文件进行计算
2. 依据字符串计算散列值的原理，将文件以字节数组的方式读入程序
3. 提供直观的应用界面，提高用户使用体验
4. 支持对不同文件类型的计算散列值
5. 添加SHA-256的算法来避免md5的碰撞性，，提高检测准确性
6. 将计算的散列值全部输出到界面，并以弹窗的方式告知检测结果

**三、课题相关数据结构及算法设计**

1主要类与函数

public class MD5Util extends JFrame { //软件Jframe界面

public MD5Util(){

super("MD5"); //为界面命名

setSize(600,500); //设置界面大小

setDefaultCloseOperation(EXIT\_ON\_CLOSE); //给界面添加窗口关闭选项

}

public static String md5HashCode(String filePath) throws FileNotFoundException{

FileInputStream fis = new FileInputStream(filePath); //将文件用流的方式读入

return md5HashCode(fis);

}

public static String md5HashCode(InputStream fis) //计算文件md5的函数

public static String sha256HashCode(InputStream fis) //计算文件SHA-256的函数

MessageDigest md = MessageDigest.getInstance(""); //调用java中的MessageDigest类

BigInteger bigInt = new BigInteger(1, md5Bytes); //调用javaBigIntel类对数取绝对值

byte[] buffer = new byte[1024]; //由于文件较大，分多次将文件读入

int length = -1;

while ((length = fis.read(buffer, 0, 1024)) != -1) {

md.update(buffer, 0, length);

2 主要算法流程

（1）MD5原理图如图2所示

N\*512bit

L bit

消息长度（L mod 2^64）

填充（0-511bit）

100...00

信息

512bit 512bit 512bit

YN-1

Y0 Y1

512 512 512

Hmd5

Hmd5

最终结果散列值

Hmd5

初始序列128bit 第一个分块的出的128bit

第N-1个128bit

图2 MD5原理图

（2）文件读入算法

文件读写可以采用以下三种方式：字节流方式、字符流方式以及缓冲区方式。其中字节流主要由InputStream和OutputStream作为基类，而字符流则主要由Reader和Writer作为基类。本次文件读入则采用了字节流方式读取文件数据。首先是创建文件流：使用文件的绝对路径直接创建：FileInputStream fis = **new** FileInputStream(filePath);也可以使用File类创建：File file = new File(filePath);FileInputStream fis = **new**FileInputStream(file);本次采用的是使用绝对路径进行直接创建。其次是读取文件流：通过FileInputStream类的read()方法读取文件流；fis.read(content);其中，content的类型是byte[]，读取到的数据都保存在该变量中。最后是关闭文件流：通过FileInputStream类的close()方法关闭文件数据流，即fis.close();流程图如图3所示

开始

创建文件流

读取文件流

关闭文件流

结束

图3 文件读入流图

（3）SHA-256散列值算法：

散列值算法是调用java中的java.security.Messagedigest类，此 MessageDigest 类为应用程序提供信息摘要算法的功能，如 MD5 或 SHA 算法。首先，创建具有制定算法名称的MessageDigest对象：[MessageDigest](http://hubingforever.blog.163.com/blog/static/171040579201210781650340/)([String](http://hubingforever.blog.163.com/blog/static/171040579201210781650340/" \o "java.lang 中的类) algorithm)；其次，调用MessageDigest类中的getInstance获取摘要算法MessageDigest.getInstance("SHA-256")；然后，向MessageDigest传送要计算的数据，通过一次或多次调用update();来完成。再次，调用digest();来计算摘要，生成散列值。最后，处理计算结果，转换为16进制：bigInt.toString(16)。SHA-256算法流程图如图4所示

处理计算结果，转换为16进制

开始

创建MessageDigest对象

调用该类中的getInstance获取摘要算法SHA-256

调用update();传送要计算的数据

调用digest();来计算摘要

输出散列值

结束

图4 散列流程图

**四、源程序代码**

import java.awt.Color;

import java.awt.Container;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.InputStream;

import java.math.BigInteger;

import java.security.MessageDigest;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.swing.JButton;

import javax.swing.JComboBox;

import javax.swing.JFileChooser;

import javax.swing.JFrame;

import static javax.swing.JFrame.EXIT\_ON\_CLOSE;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import javax.swing.JTextArea;

import javax.swing.JTextField;

import javax.swing.filechooser.FileNameExtensionFilter;

public class MD5Util extends JFrame {

String s; //前文件路径

String w; //后文件路径

public MD5Util(){

super("文件完整性检测");

setSize(600,500);

setDefaultCloseOperation(EXIT\_ON\_CLOSE);

}

public static void main(String[] args) {

MD5Util md5=new MD5Util();

Container contentPane = md5.getContentPane();

JPanel j=new JPanel();

j.setBackground(Color.white);

JPanel j1=new JPanel();

j1.setBackground(Color.white);

md5.setLayout(null);

j.setBounds(0, 100, 600, 50);

j1.setBounds(150, 180, 200, 50);

JLabel label1=new JLabel("输入原文件路径");

JLabel label2=new JLabel("输入后文件路径");

JButton b=new JButton("检测");

JButton b1=new JButton("浏览");

JButton b2=new JButton("浏览");

j.add(b1);

j.add(b2);

j1.add(b);

j.add(label1);

j.add(label2);

JTextField jf1 =new JTextField(400);

JTextField jf2 =new JTextField(400);

JTextArea jf3 =new JTextArea();

j.add(jf1);

j.add(jf2);

j.add(jf3);

j.setLayout(null);

label1.setBounds(0, 10, 100, 20);

label2.setBounds(0, 30, 100, 20);

b1.setBounds(500, 10, 100, 20);

b2.setBounds(500, 30, 100, 20);

jf1.setBounds(110, 10, 300, 20);

jf2.setBounds(110, 30, 300, 20);

jf3.setBounds(0, 230, 600, 200);

md5.add(j);

md5.add(j1);

md5.add(jf3);

b1.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) { //按钮点击事件

JFileChooser chooser = new JFileChooser("C:\\Users\\w\\Desktop"); //设置选择器

int returnVal = chooser.showOpenDialog(b1); //是否打开文件选择框

if (returnVal == JFileChooser.APPROVE\_OPTION) { //如果符合文件类型

md5.s = chooser.getSelectedFile().getAbsolutePath(); //获取绝对路径

jf1.setText(md5.s);

}

}

});

b2.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) { //按钮点击事件

JFileChooser chooser = new JFileChooser("C:\\Users\\w\\Desktop"); //设置选择器

int returnVal = chooser.showOpenDialog(b2); //是否打开文件选择框

if (returnVal == JFileChooser.APPROVE\_OPTION) { //如果符合文件类型

md5.w = chooser.getSelectedFile().getAbsolutePath(); //获取绝对路径

jf2.setText(md5.w);

}

}

});

b.addActionListener(new ActionListener(){

@Override

public void actionPerformed(ActionEvent e) {

if(jf1.getText().trim().equals("")||jf2.getText().trim().equals("")){

JOptionPane.showMessageDialog(null, "请输入两个对比文件路径", "检测", JOptionPane.ERROR\_MESSAGE);

}

else{

try {

md5.s=jf1.getText();

String s1=md5HashCode(md5.s);

String s2=sha256HashCode(md5.s);

md5.w=jf2.getText();

String w1=md5HashCode(md5.w);

String w2=sha256HashCode(md5.w);

jf3.setText(s1+"\t源文件MD5值\n"+w1+"\t后文件MD5值\n"+s2+"\t源文件SHA-256值\n"+w2+"\t后文件SHA-256值");

if(s1.equals(w1)&&s2.equals(w2)){

JOptionPane.showMessageDialog(null, "文件未被修改", "检测结果", JOptionPane.INFORMATION\_MESSAGE);

}

else{

JOptionPane.showMessageDialog(null, "文件已被修改", "检测结果", JOptionPane.WARNING\_MESSAGE);

}

} catch (FileNotFoundException ex) {

Logger.getLogger(MD5Util.class.getName()).log(Level.SEVERE, null, ex);

}

}

}

});

md5.setVisible(true);

}

public static String md5HashCode(String filePath) throws FileNotFoundException{

FileInputStream fis = new FileInputStream(filePath);

return md5HashCode(fis);

}

public static String md5HashCode(InputStream fis) {

try {

//拿到一个MD5转换器,如果想使用SHA-1或SHA-256，则传入SHA-1,SHA-256

MessageDigest md = MessageDigest.getInstance("MD5");

//分多次将一个文件读入，对于大型文件而言，比较推荐这种方式，占用内存比较少。

byte[] buffer = new byte[1024];

int length = -1;

while ((length = fis.read(buffer, 0, 1024)) != -1) {

md.update(buffer, 0, length);

}

fis.close();

//转换并返回包含16个元素字节数组,返回数值范围为-128到127

byte[] md5Bytes = md.digest();

BigInteger bigInt = new BigInteger(1, md5Bytes); //1代表绝对值

return bigInt.toString(16); //转换为16进制

} catch (Exception e) {

e.printStackTrace();

return "";

}

}

public static String sha256HashCode(String filePath) throws FileNotFoundException{

FileInputStream fis = new FileInputStream(filePath);

return sha256HashCode(fis);

}

public static String sha256HashCode(InputStream fis) {

try {

//拿到一个MD5转换器,如果想使用SHA-1或SHA-256，则传入SHA-1,SHA-256

MessageDigest sha256 = MessageDigest.getInstance("SHA-256");

//分多次将一个文件读入，对于大型文件而言，比较推荐这种方式，占用内存比较少。

byte[] buffer = new byte[1024];

int length = -1;

while ((length = fis.read(buffer, 0, 1024)) != -1) {

sha256.update(buffer, 0, length);

}

fis.close();

//转换并返回包含16个元素字节数组,返回数值范围为-128到127

byte[] sha256Bytes = sha256.digest();

BigInteger bigInt = new BigInteger(1, sha256Bytes);//1代表绝对值

return bigInt.toString(16);//转换为16进制

} catch (Exception e) {

e.printStackTrace();

return "";

}

}

}

**五、测试数据及其结果分析**

1. 对算法功能、性能以及健壮性的评测分析：

1. 用户可选择两个文件的路径用于检测，当用户没有选择两个路径时会弹窗提示，结果如图5所示

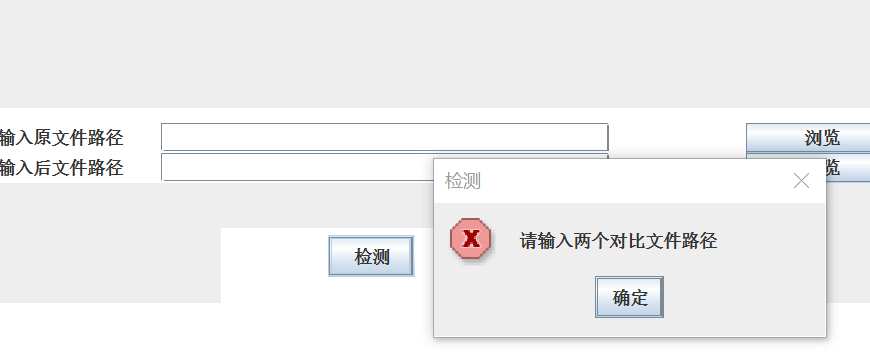


图5 未选择路径弹窗图

1. 如图6所示，我们选择两个不同路径下地相同文本文件“文件完整性.txt”对比两个文件的MD5和SHA-256的散列值，结果如图6所示

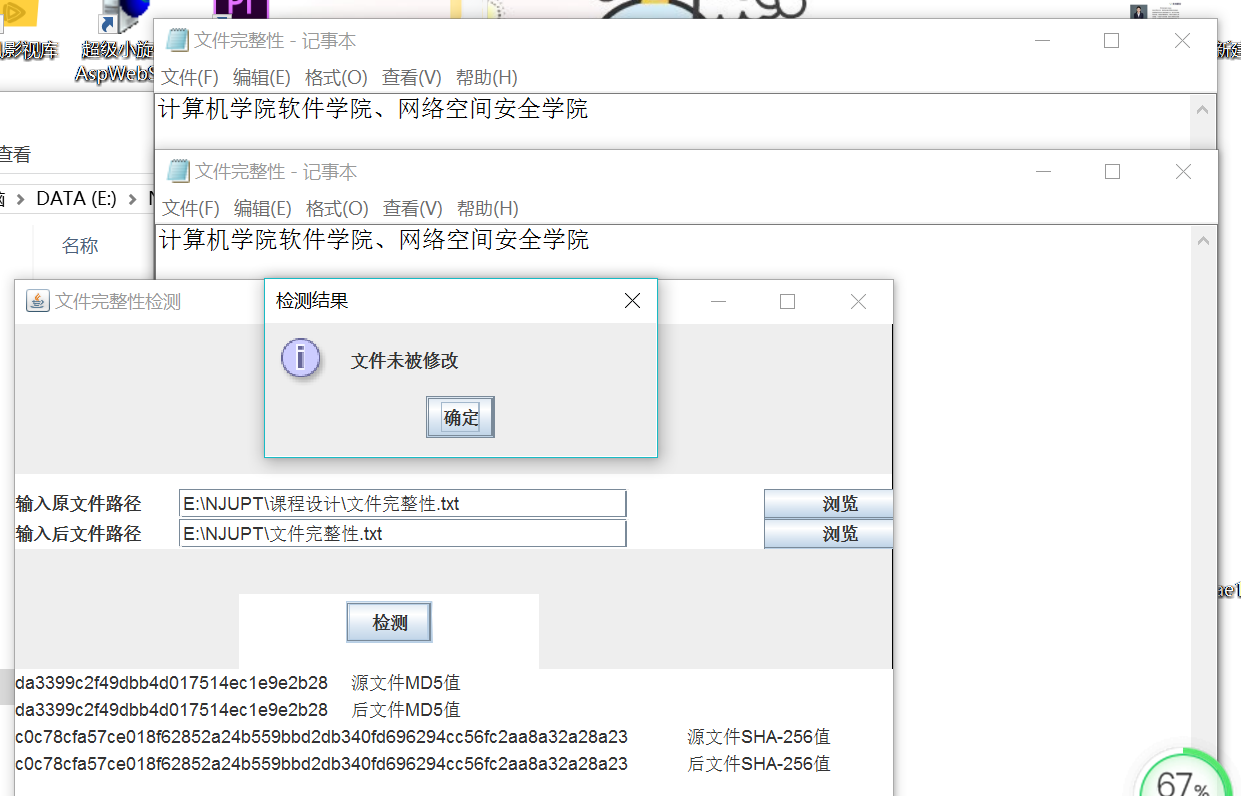


图6 检测相同文本文件结果图

1. 同理，修改其中一个文件内容重新检测，检测结果如图7所示

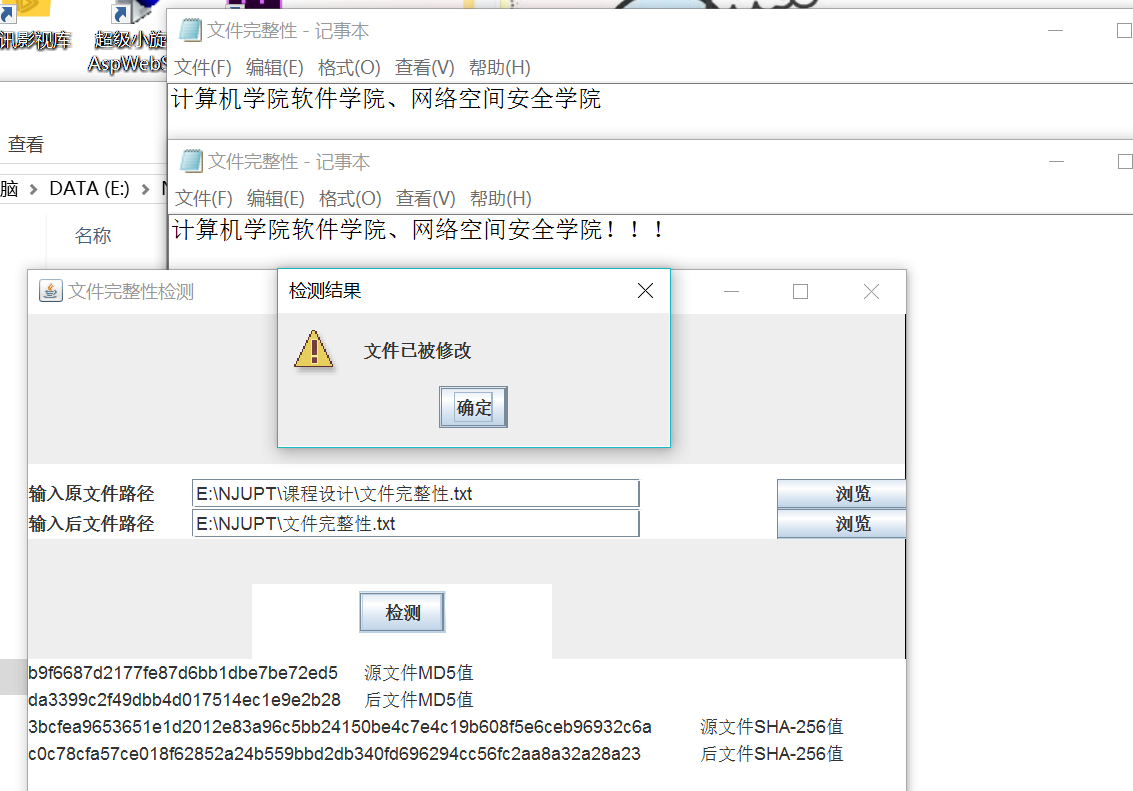


图7 被修改后的结果图

1. 检测.cpp类型文件，检测相同的.cpp文件，结果如图8所示



图8 检测相同内容的.cpp文件结果图

1. 同理修改其中一个.cpp文件内容，重新检测，检测结果如图9所示

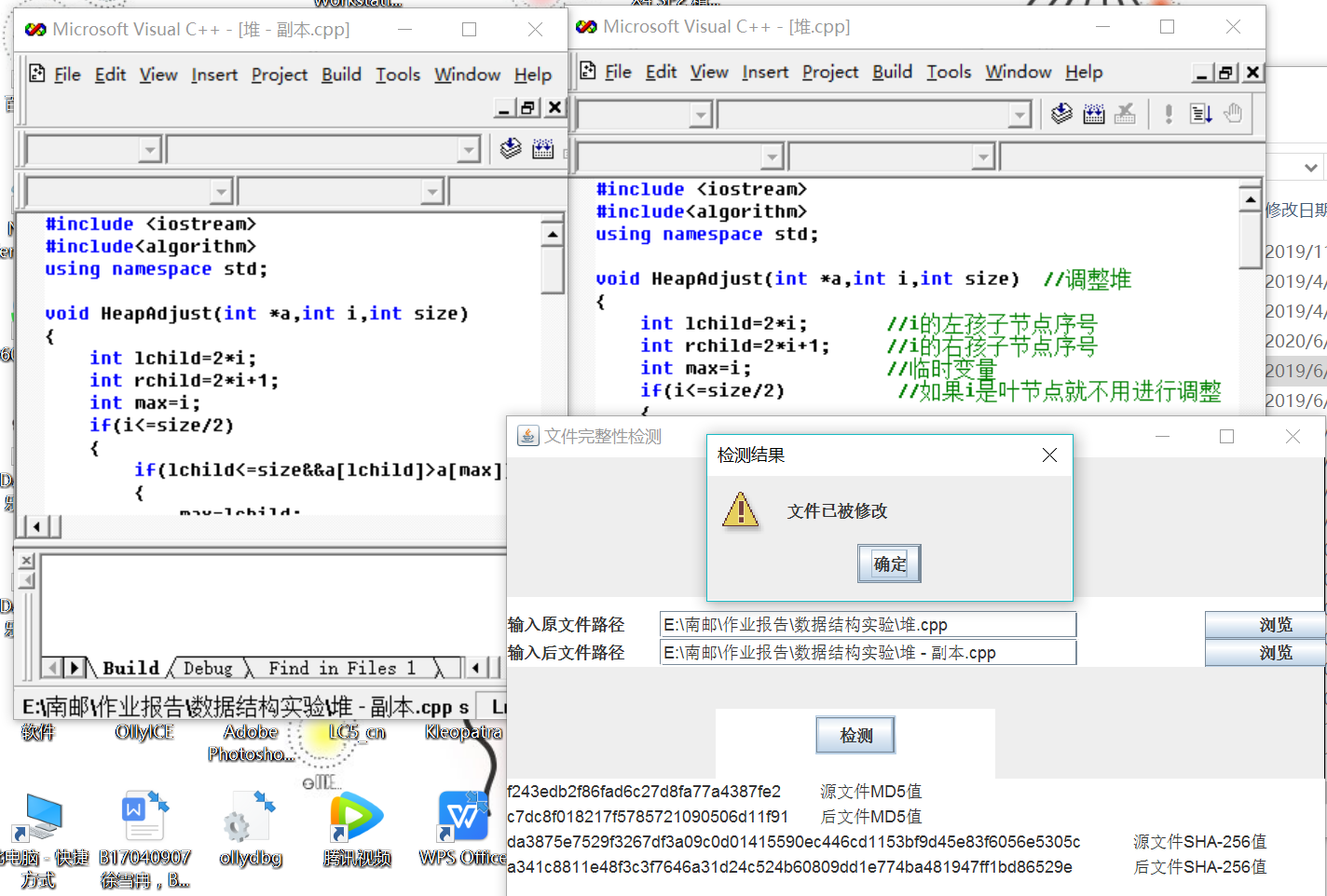


图9 修改后的结果图

1. 综上所述，该软件通过计算文件的散列值可以实现对不同文件完整性的检测。即到达题目要求中的文件可能因为感染病毒、植入木马/后门、人为篡改或者传输故障等原因导致被篡改，从而能够及时检测。但是，该软件存在无法对同一文件比如.exe文件进行完整性检测，存在一些问题。

**六、课题完成过程中遇到的问题及解决方法**

问题1：怎么讲文件读入，从而可以进行散列值计算？

解决方法：查阅资料得知文件读写可以采用以下三种方式：字节流方式、字符流方式以及缓冲区方式。其中字节流主要由InputStream和OutputStream作为基类，而字符流则主要由Reader和Writer作为基类。而此次主要使用字节流输出的方式，将文件转为字节数组，这样就将为题转化为了和字符串一样了，在这里调用java中的FileInputStream函数

问题2：在调用java的messagedigest类方法的时候，有时候文件很大，将文件字节数组读入转换器时要占用很多内存空间。

解决方法：分多次将一个文件读入，对于大型文件而言，占用内存比较少。

问题3：未将摘要结果进行处理转化为16进制。

解决方法：直接利用BigInteger .toString(16)进行快速转换。之所以要转化为16进制，是因为byte[]中，每个元素都占8个二进制位，每个位都有两种可能，于是16进制中每两位对应byte中的一位。比如byte中第3位的值为1，那么这一位的值就会转成16进制中的5了。

问题4：在java中转换器返回的数值范围是-128到127，是得计算后的SHA-256散列值并不正确。

解决方法：使用BigInteger(1, md5Bytes)函数将结果转为绝对值即可。

问题5：软件大体功能能够实现，就是用户使用体验很差。

解决方法：使用java界面中的组件，如Jframe，对话框，弹窗等，提高用户体验。

**七、总结**

这次的课程设计，使我能够将之前课程所学的知识落到实践中。让我对文件完整性检测有了进一步的认识。大部分计算机系统的核心是文件系统，文件系统是比较普遍的被攻击的目标，入侵者通常修改系统配置和程序以便于下次进入系统。通常修改系统日志来掩盖他们的入侵路径和阻碍入侵的发现，用户文件的修改和破坏将会损坏安全策略。因此需要对文件进行完整性检测。由于MD5具有抗修改性：即便修改一个字节，计算出来的MD5值也会巨大差异，所以每个文件都可以用MD5验证程序算出一个固定的MD5值，是独一无二的。一般来说，开发方会在软件发布时预先算出文件的MD5值，如果文件被盗用，加了木马或者被篡改版权，那么它的MD5值也随之改变，也就是说我们对比文件当前的MD5值和它标准的MD5值来检验它是否正确和完整。但由于MD5具有抗弱碰撞性，所以本次实验增加里SHA-256算法来提高数据检测的准确性。

此外，在此次课程设计还学会了运用各种库函数。对消息摘要也有了进一步了解，巧用java中的MessageDigest类 ，该类用于为应用程序提供信息摘要算法的功能，如 MD5 或 SHA 算法。也就是用于生成散列码。信息摘要是安全的单向哈希函数，它接收任意大小的数据，输出固定长度的哈希值。MessageDigest 通过其getInstance系列静态函数来进行实例化和初始化。MessageDigest 对象通过使用[update](http://hubingforever.blog.163.com/blog/static/171040579201210781650340/)（）方法处理数据。任何时候都可以调用[reset](http://hubingforever.blog.163.com/blog/static/171040579201210781650340/)（）方法重置摘要。一旦所有需要更新的数据都已经被更新了，应该调用[digest](http://hubingforever.blog.163.com/blog/static/171040579201210781650340/)（）方法之一完成哈希计算并返回结果。

综上所述，此次课程设计既是对理论的完善与补充，也是对理论的实验与应用。让我更加意识信息安全的重要性。